

AD-A066 060

CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/2  
SYSTOLIC ARRAYS FOR (VLSI).(U)

DEC 78 H T KUNG, C E LEISERSON

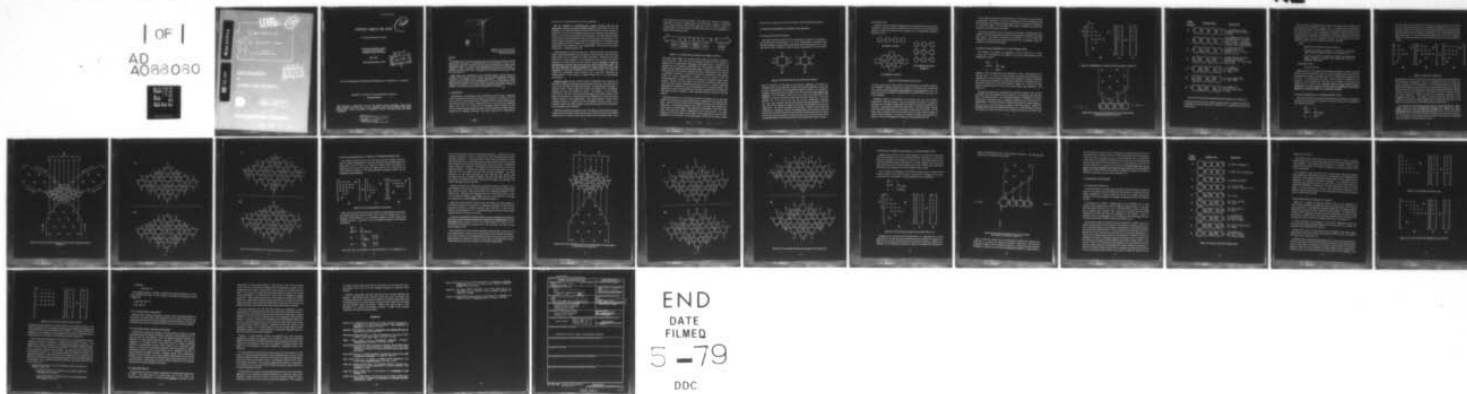
N00014-76-C-0370

UNCLASSIFIED

CMU-CS-79-103

NL

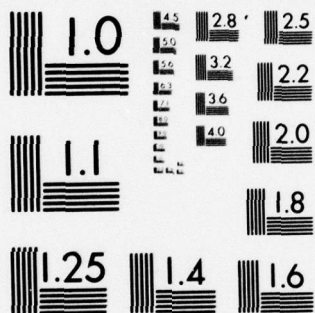
| OF |  
AD  
A088080



END  
DATE  
FILMED

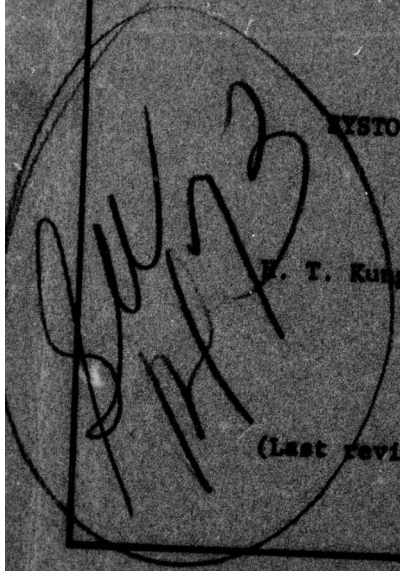
5-79

DDC



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL



SYSTOLIC ARRAYS FOR (VLSI)

H. T. Kung and Charles E. Leiserson

April 1978

(Last revised December 1978)

DEPARTMENT  
of



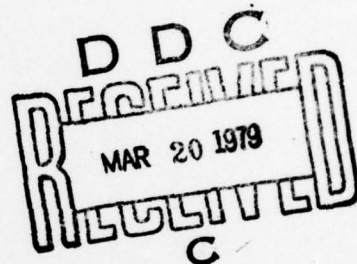
# SYSTOLIC ARRAYS FOR (VLSI)

H. T. Kung and Charles E. Leiserson

Department of Computer Science  
Carnegie-Mellon University  
Pittsburgh, Pennsylvania 15213

April 1978

(Last revised December 1978)



[In the forthcoming book Introduction to VLSI Systems by C. A. Mead and L. A. Conway]

Copyright -C- 1979 by H.T. Kung and Charles E. Leiserson

*All Rights Reserved*

This research is supported in part by the National Science Foundation under Grant MCS 75-222-55 and the Office of Naval Research under Contract N00014-76-C-0370, NR 044-422. Charles E. Leiserson is supported in part by the Fannie and John Hertz Foundation.

This document has been approved  
for public release and sale; its  
distribution is unlimited.



ACCESSION FOR	
NO.	Write Section <input checked="" type="checkbox"/>
NO.	Buff Section <input type="checkbox"/>
DATE RECEIVED	
JUL 1 1974	
NOTES FOR THE DIRECTOR	
SPECIAL	
A	

*And now I see with eye serene  
The very pulse of the machine.  
--William Wordsworth*

#### Abstract

A systolic system is a network of processors which rhythmically compute and pass data through the system. Physiologists use the word "systole" to refer to the rhythmically recurrent contraction of the heart and arteries which pulses blood through the body. In a systolic computing system, the function of a processor is analogous to that of the heart. Every processor regularly pumps data in and out, each time performing some short computation, so that a regular flow of data is kept up in the network.

Many basic matrix computations can be pipelined elegantly and efficiently on systolic networks having an array structure. As an example, hexagonally connected processors can optimally perform matrix multiplication. Surprisingly, a similar systolic array can compute the LU-decomposition of a matrix. These systolic arrays enjoy simple and regular communication paths, and almost all processors used in the networks are identical. As a result, special purpose hardware devices based on systolic arrays can be built inexpensively using the VLSI technology.

#### 1. Introduction

Developments in microelectronics have revolutionized computer design. Integrated circuit technology has increased the number and complexity of components that can fit on a chip or a printed circuit board. Component density has been doubling every one-to-two years and already, a multiplier can fit on a very large scale integrated (VLSI) circuit chip. As a result, the new technology makes it feasible to build low-cost special purpose, peripheral devices to rapidly solve sophisticated problems. Reflecting the changing technology, this paper proposes new multiprocessor

structures for processing some basic matrix computations.

We are interested in high-performance parallel structures that can be implemented directly as low-cost hardware devices. By performance, we are not referring to the traditional operation counts that characterize classical analyses of algorithms, but rather, the throughput obtainable when a special purpose peripheral device is attached to a general purpose host computer. This implies that time spent in I/O, control, and data movement as well as arithmetic must all be considered. VLSI offers excellent opportunities for inexpensive implementation of high performance devices (Mead and Conway [1978]). Thus, in this paper the cost of a device will be determined by the expense of a VLSI implementation. "Fit the job to the bargain components" -- Blakeslee [1975, p. 4].

VLSI technology has made one thing clear. Simple and regular interconnections lead to cheap implementations and high densities, and high density implies both high performance and low overhead for support components. (Sutherland and Mead [1977] has a good discussion on the importance of having simple and regular geometries for data paths.) For these reasons, we are interested in designing multiprocessor structures which have simple and regular communication paths. We are also interested in employing pipelining as a general method for using these structures. By pipelining, computation may proceed concurrently with input and output, and consequently overall execution time is minimized. Pipelining plus multiprocessing at each stage of a pipeline should lead to the best-possible performance.

Systolic systems provide a realistic model of computation which captures the concepts of pipelining, parallelism and interconnection structures. We do not want to give a formal definition of systolic systems here. For the purpose of this paper, it suffices to view a systolic system as a network of processors which rhythmically compute and pass data through the system. The analogy is to the rhythmic contraction of the heart which pulses blood through the circulatory system of the body. Each processor in a systolic network can be thought of as a heart that pumps multiple streams of data through itself. The regular beating of these parallel processors keeps up a constant flow of data throughout the entire network. As a processor pumps data items through, it performs some constant-time computation and may update some of the items.

Unlike the closed-loop circulatory system of the body, a systolic computing system usually has ports into which inputs flow, and ports where the results of the systolic computation are retrieved. Thus a systolic system can be a pipelined system - input

and output occur with every pulsation. This makes them attractive as peripheral processors attached to the data channel of a host computer. Figure 1-1 illustrates how a special purpose systolic device might form a part of a PDP-11 system. A systolic device may also process a real-time data stream or be a component in a larger special purpose system.

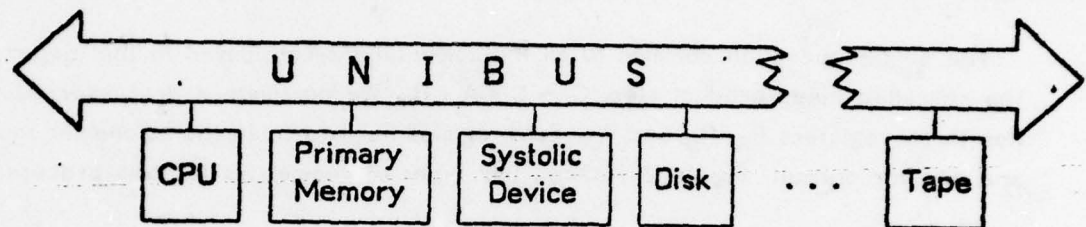


Figure 1-1: A systolic device connected to the UNIBUS of a PDP-11.

This paper deals largely with systolic systems where the underlying network is array structured. (See also Kung and Leiserson [1978].) An array network is attractive for it enjoys simple and regular communication paths. In Section 2, we describe the basic hardware requirements and interconnection schemes for the systolic arrays proposed and discuss the feasibility of building them in VLSI. Section 3 deals with the matrix-vector multiplication problem. Multiplication of two matrices is considered in Section 4. In Section 5, we show that essentially the same systolic arrays for matrix multiplication in Section 4 can be used to find the LU-decomposition of a matrix. Section 6 is concerned with solving triangular linear systems. We show that this problem can be solved by almost the same systolic array for matrix-vector multiplication described in Section 3. Section 7 discusses applications and extensions of the results presented in the previous sections. The applications include the computations of finite impulse response filters, convolutions, and discrete Fourier transforms. Some concluding remarks are given in the last section.

The size of each of our systolic array networks is dependent only on the band width of the band matrix to be processed, and is independent of the length of the band. Thus, a fixed size systolic array can pipeline band matrices with arbitrarily long bands. The pipelining aspect of our arrays is, of course, most effective for band matrices with long bands. Band matrices are interesting in their own right, since many important scientific computations involve band matrices. For these reasons, most of the results in this paper will be presented in terms of their applications to band matrices. All the results apply to dense matrices since a dense



matrix can be viewed as a band matrix having the maximum-possible band width.

## 2. The Basic Components and Systolic Array Structures

### 2.1 The Inner Product Step Processor

The single operation common to all the computations considered in this paper is the so-called inner product step,  $C \leftarrow C + A \times B$ . We postulate a processor which has three registers  $R_A$ ,  $R_B$ , and  $R_C$ . Each register has two connections, one for input and one for output. Figure 2-1 shows two types of geometries for this processor.

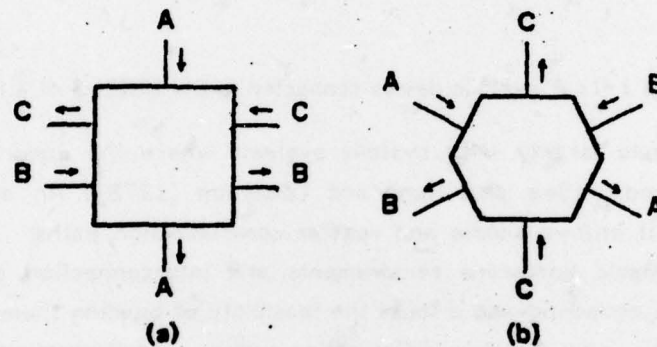


Figure 2-1: Geometries for the inner product step processor.

Type (a) geometry will be used for matrix-vector multiplication and solution of triangular linear systems (Sections 3 and 6), whereas type (b) geometry will be used for matrix multiplication and LU-decomposition (Sections 4 and 5). The processor is capable of performing the inner product step and is called the inner product step processor. We shall define a basic time unit in terms of the operation of this processor. In each unit time interval, the processor shifts the data on its input lines denoted by A, B and C into  $R_A$ ,  $R_B$  and  $R_C$ , respectively, computes  $R_C \leftarrow R_C + R_A \times R_B$ , and makes the input values for  $R_A$  and  $R_B$  together with the new value of  $R_C$  available as outputs on the output lines denoted by A, B and C, respectively. All outputs are latched and the logic is clocked so that when one processor is connected to another, the changing output of one during a unit time interval will not interfere with the input to another during this time interval. This is not the only processing element we shall make use of, but it will be the work horse. A special processor for performing division will be specified later when it is used.

## 2.2 Systolic Arrays

A systolic device is typically composed of many interconnected inner product step processors. The basic network organization we shall adopt is the mesh-connected scheme in which all connections from a processor are to neighboring processors. (See Figure 2-2.)

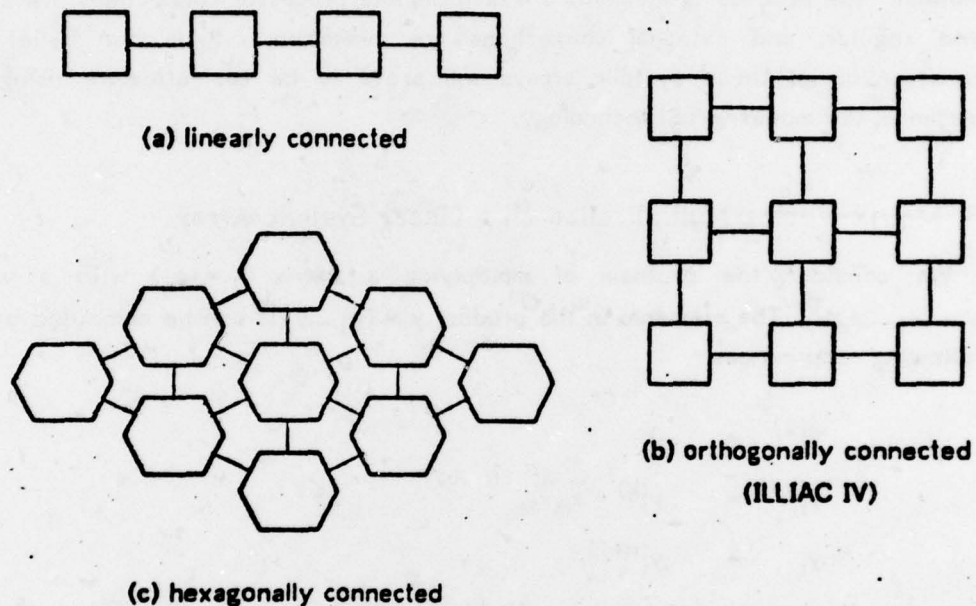


Figure 2-2: Mesh-connected systolic arrays.

The most widely known system based on this organization is the ILLIAC IV (Barnes et al. [1968]). If diagonal connections are added in one direction only, we shall call the resulting scheme hexagonally mesh-connected or hex-connected for short. We shall demonstrate that linearly connected and hex-connected arrays are natural for matrix problems.

Processors lying on the boundary of the systolic array may have external connections to the host memory. Thus, an input/output data path of a boundary processor may sometimes be designated as an external input/output connection for the device. A boundary processor may receive input from the host memory through such an external connection, or it may receive a fixed value such as zero. On the other hand, a boundary processor can send data to the host memory through an external output connection. An output of a boundary processor may sometimes be ignored. This will be designated by omitting the corresponding output line.

In this paper we assume that the processors in a systolic array are synchronous as described in Section 2.1. However, it is possible to view the processors being asynchronous, each computing its output values when all its inputs are available, as in a data flow model. For the results of this paper we believe the synchronous approach to be more direct and intuitive.

The hardware demands of the systolic arrays in this paper are readily seen to be modest. The processing elements are uniform, interprocessor connections are simple and regular, and external connections are minimized. It is our belief that construction of these systolic arrays will prove to be cost-effective using, for instance, the modern VLSI technology.

### 3. Matrix-Vector Multiplication on a Linear Systolic Array

We consider the problem of multiplying a matrix  $A = (a_{ij})$  with a vector  $x = (x_1, \dots, x_n)^T$ . The elements in the product  $y = (y_1, \dots, y_n)^T$  can be computed by the following recurrences.

$$\begin{aligned} y_i^{(1)} &= 0, \\ y_i^{(k+1)} &= y_i^{(k)} + a_{ik}x_k, \\ y_i &= y_i^{(n+1)}. \end{aligned}$$

Suppose  $A$  is an  $n \times n$  band matrix with band width  $w = p+q-1$ . (See Figure 3-1 for the case when  $p = 2$  and  $q = 3$ .) Then the above recurrences can be evaluated by pipelining the  $x_i$  and  $y_i$  through a systolic array consisting of  $w$  linearly connected inner product step processors. We illustrate the operation of the systolic array for the band matrix-vector multiplication problem in Figure 3-1. For this case the linearly connected systolic array has four inner product step processors. See Figure 3-2.

The general scheme of the computation can be viewed as follows. The  $y_i$ , which are initially zero, are pumped to the left while the  $x_i$  are pumped to the right and the  $a_{ij}$  are marching down. (For the general problem of computing  $Ax+d$  where  $d=(d_1, \dots, d_n)^T$  is any given vector,  $y_i$  should be initialized as  $d_i$ .) All the moves are synchronized. It turns out that each  $y_i$  is able to accumulate all its terms, namely,  $a_{ii}x_i$ ,  $a_{i,i-1}x_{i-1}$ ,  $a_{i,i+1}x_{i+1}$ , before it leaves the network. Figure 3-3 illustrates the first seven pulsations of the systolic array. Note that when  $y_1$  and  $y_2$  are output they have the correct values. Observe also that at any given time



$$\begin{bmatrix}
 & \overbrace{\quad}^p & & & & \\
 \underbrace{\quad}_q & a_{11} & a_{12} & & & \\
 & a_{21} & a_{22} & a_{23} & & \\
 & a_{31} & a_{32} & a_{33} & a_{34} & \\
 & & a_{42} & a_{43} & a_{44} & a_{45} \\
 & & & a_{53} & & \\
 & & & & 0 & \\
 & & & & & 0
 \end{bmatrix}
 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

$A \qquad \qquad \qquad x \qquad \qquad \qquad y$

Figure 3-1: Multiplication of a vector by a band matrix with  $p = 2$  and  $q = 3$ .

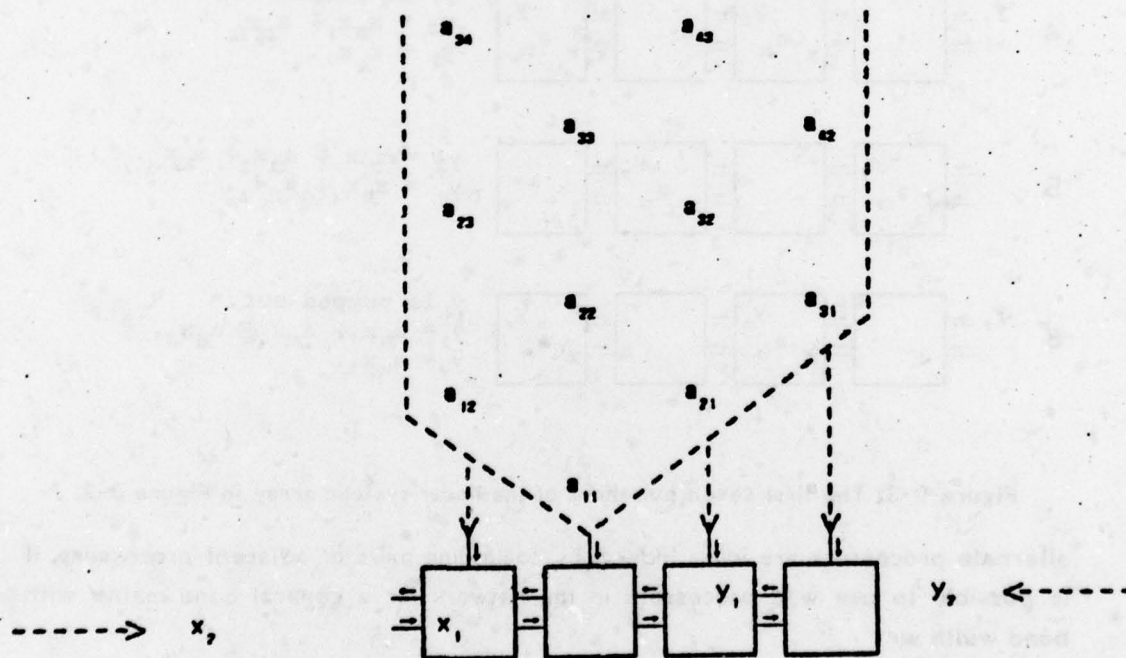


Figure 3-2: The linearly connected systolic array for the matrix-vector multiplication problem in Figure 3-1.



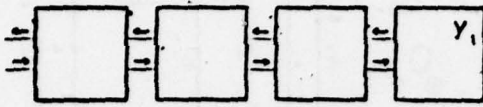
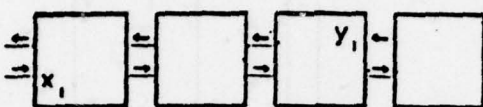
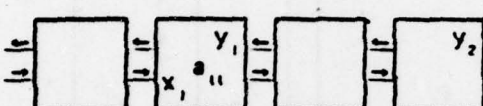
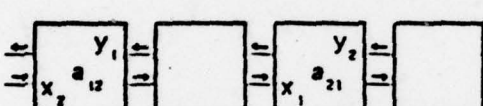
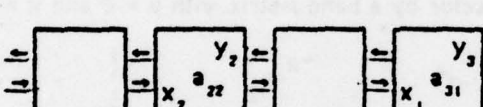
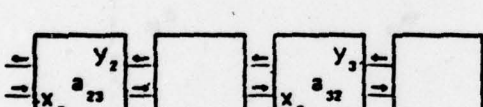
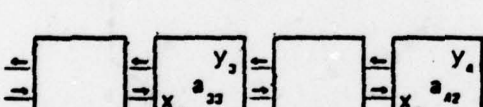
Pulse Number	Configuration	Comments
0		$y_1$ , initialized as zero, is pumped into the fourth processor.
1		$x_1$ is pumped into the first processor while $y_1$ is moved left one place. (From now on the $x_i$ and $y_i$ keep moving right and left, respectively.)
2		$a_{11}$ enters the second processor where $y_1$ is updated by $y_1 \leftarrow y_1 + a_{11} x_1$ . Thus $y_1 = a_{11} x_1$ .
3		$a_{12}$ and $a_{21}$ enter the first and third processors, respectively. $y_1 = a_{11} x_1 + a_{12} x_2$ and $y_2 = a_{21} x_1$ .
4		$y_1$ is pumped out. $y_2 = a_{21} x_1 + a_{22} x_2$ $y_3 = a_{31} x_1$ .
5		$y_2 = a_{21} x_1 + a_{22} x_2 + a_{23} x_3$ $y_3 = a_{31} x_1 + a_{32} x_2$ .
6		$y_2$ is pumped out. $y_3 = a_{31} x_1 + a_{32} x_2 + a_{33} x_3$ $y_4 = a_{42} x_2$ .

Figure 3-3: The first seven pulsations of the linear systolic array in Figure 3-2.

alternate processors are idle. Indeed, by coalescing pairs of adjacent processors, it is possible to use  $w/2$  processors in the network for a general band matrix with band width  $w$ .

We now specify the operation of the systolic array more precisely. Assume that the processors are numbered by integers  $1, 2, \dots, w$  from the left end processor to the right end processor. Each processor has three registers,  $R_A$ ,  $R_x$  and  $R_y$ , which will hold entries in  $A$ ,  $x$  and  $y$ , respectively. Initially, all registers contain zeros. Each pulsation of the systolic array consists of the following operations, but for odd numbered pulses only odd numbered processors are activated and for even numbered pulses only even numbered processors are activated.

#### 1. Shift.

- $R_A$  gets a new element in the band of matrix  $A$ .
- $R_x$  gets the contents of register  $R_x$  from the left neighboring node. (The  $R_x$  in processor 1 gets a new component of  $x$ .)
- $R_y$  gets the contents of register  $R_y$  from the right neighboring node. (Processor 1 outputs its  $R_y$  contents and the  $R_y$  in processor  $w$  gets zero.)

#### 2. Multiply and Add.

$$R_y \leftarrow R_y + R_A \times R_x.$$

Using the type (a) inner product step processor postulated in section 2, we note that the three shift operations in step 1 can be done simultaneously, and that each pulsation of the systolic array takes a unit of time. Suppose the bandwidth of  $A$  is  $w = p+q-1$ . It is readily seen that after  $w$  units of time the components of the product  $y = Ax$  are pumped out from the left end processor at the rate of one output every two units of time. Therefore, using our systolic network all the  $n$  components of  $y$  can be computed in  $2n+w$  time units, as compared to the  $O(nw)$  time needed for a sequential algorithm on a uniprocessor computer.

### 4. Matrix Multiplication on a Hexagonal Systolic Array

This section considers the problem of multiplying two  $n \times n$  matrices. It is easy to see that the matrix product  $C = (c_{ij})$  of  $A = (a_{ij})$  and  $B = (b_{ij})$  can be computed by the following recurrences.

$$\begin{aligned} c_{ij}^{(1)} &= 0, \\ c_{ij}^{(k+1)} &= c_{ij}^{(k)} + a_{ik}b_{kj}, \\ c_{ij} &= c_{ij}^{(n+1)}. \end{aligned}$$

Let A and B be  $n \times n$  band matrices of band width  $w_1$  and  $w_2$ , respectively. We show how the recurrences above can be evaluated by pipelining the  $a_{ij}$ ,  $b_{ij}$  and  $c_{ij}$  through a systolic array having  $w_1 w_2$  hex-connected inner product step processors. We illustrate the general scheme by considering the matrix multiplication problem depicted in Figure 4-1. The diamond shaped systolic array for this case is shown in Figure 4-2, where processors are hex-connected and data flows are indicated by arrows.

$$\begin{array}{ccc}
 \begin{bmatrix} a_{11} & a_{12} & & & 0 \\ & a_{21} & a_{22} & a_{23} & \\ & a_{31} & a_{32} & a_{33} & a_{34} \\ & & a_{42} & & \\ 0 & & & & \ddots \end{bmatrix} & \begin{bmatrix} b_{11} & b_{12} & b_{13} & & 0 \\ & b_{21} & b_{22} & b_{23} & b_{24} \\ & & b_{32} & b_{33} & b_{34} & b_{35} \\ & & & b_{42} & & \\ 0 & & & & & \ddots \end{bmatrix} & = & \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & 0 \\ & c_{21} & c_{22} & c_{23} & c_{24} \\ & & c_{31} & c_{32} & c_{33} & c_{34} \\ & & & c_{41} & c_{42} & \\ 0 & & & & & \ddots \end{bmatrix} \\
 A & B & & C
 \end{array}$$

Figure 4-1: Band matrix multiplication.

The elements in the bands of A, B and C are pumped through the systolic network in three directions synchronously. Each  $c_{ij}$  is initialized to zero as it enters the network through the bottom boundaries. (For the general problem of computing  $AB+D$  where  $D=(d_{ij})$  is any given matrix,  $c_{ij}$  should be initialized as  $d_{ij}$ .) One can easily see that with the type (b) inner product step processors described in Section 2, each  $c_{ij}$  is able to accumulate all its terms before it leaves the network through the upper boundaries. Figure 4-3 shows four consecutive pulsations of the hexagonal systolic array. The reader is invited to study the data flow of this problem more closely by making transparencies of the band matrices shown in the figures, and moving them over the network picture as described above.

Let A and B be  $n \times n$  band matrices of band width  $w_1$  and  $w_2$ , respectively. Then a systolic array of  $w_1 w_2$  hex-connected processors can pipeline the matrix multiplication  $A \times B$  in  $3n + \min(w_1, w_2)$  units of time. Note that in any row or column of the network, out of every three consecutive processors, only one is active at given time. It is possible to use about  $(w_1 w_2)/3$  processors in the network for multiplying two band matrices with band widths  $w_1$  and  $w_2$ .



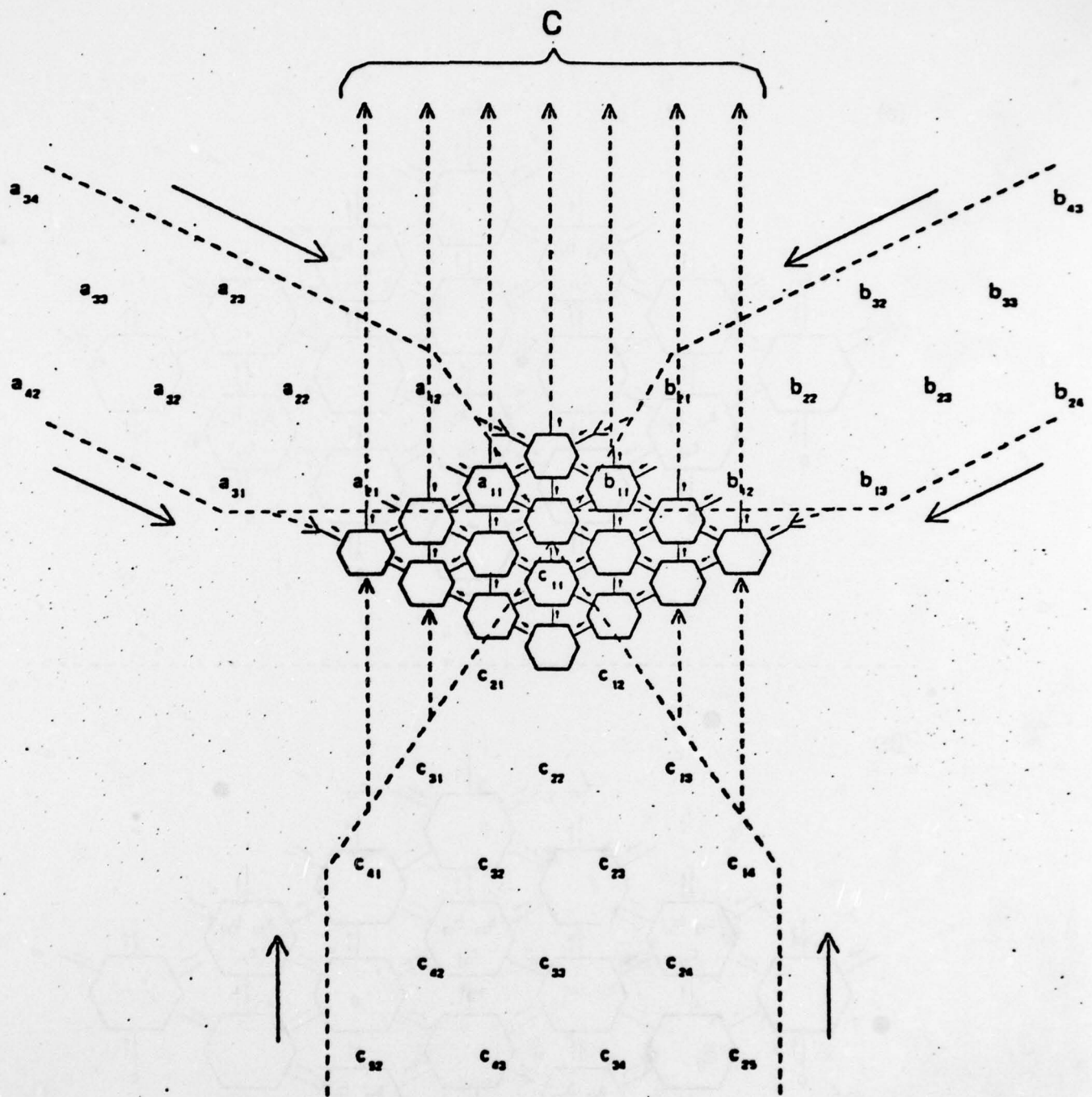
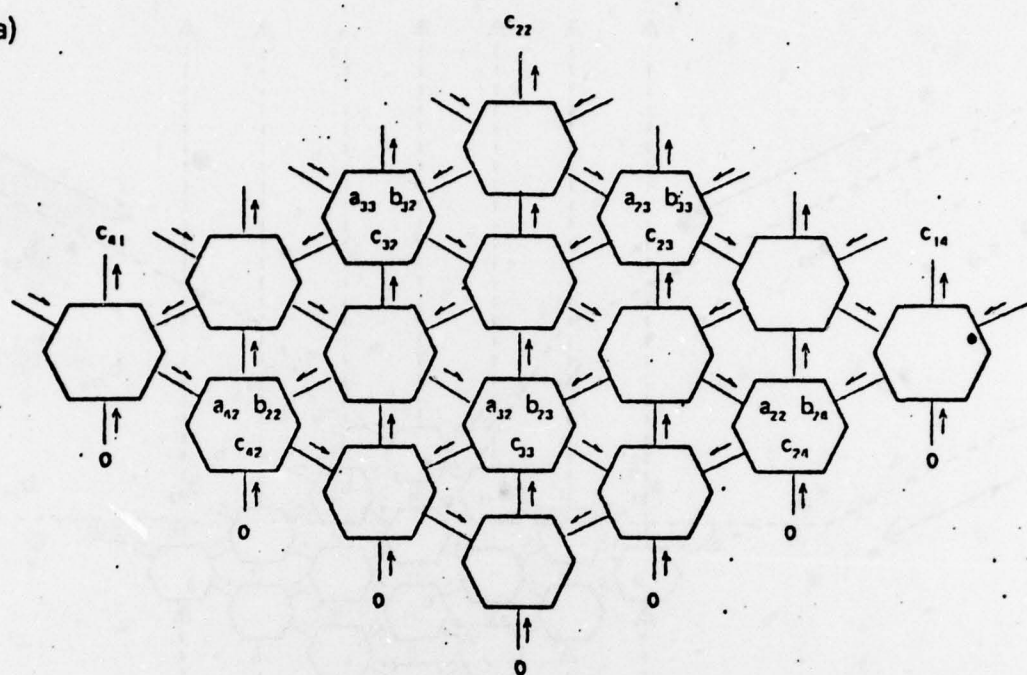
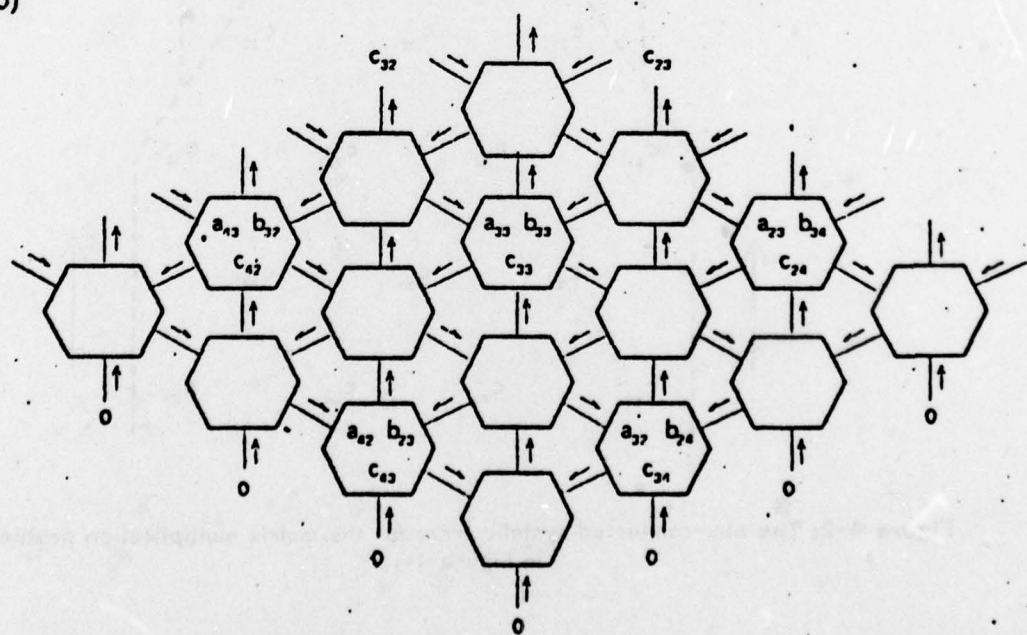


Figure 4-2: The hex-connected systolic array for the matrix multiplication problem in Figure 4-1.

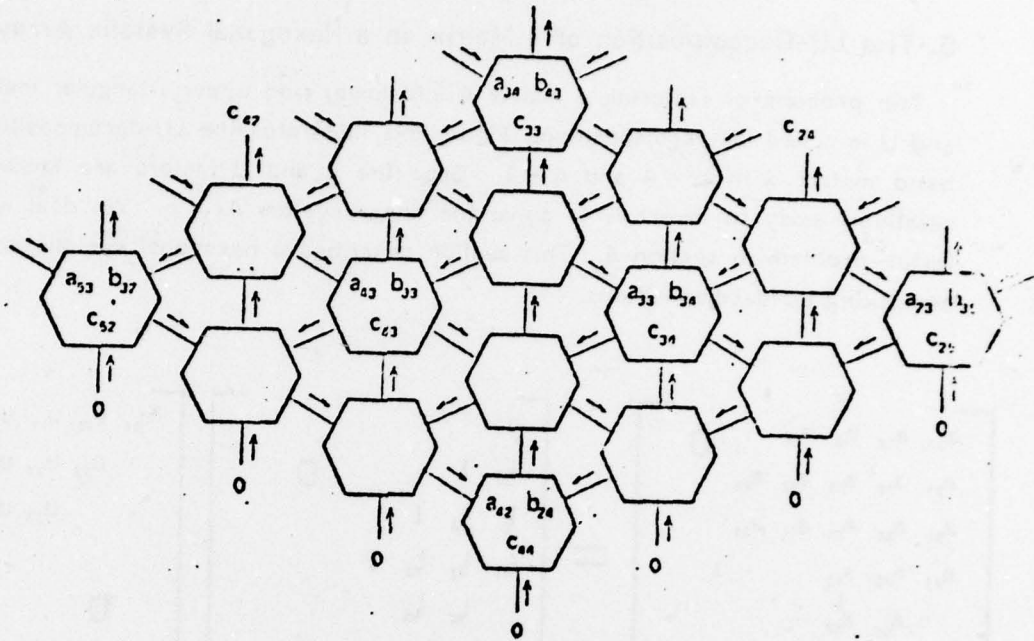
(a)



(b)



(c)



(d)

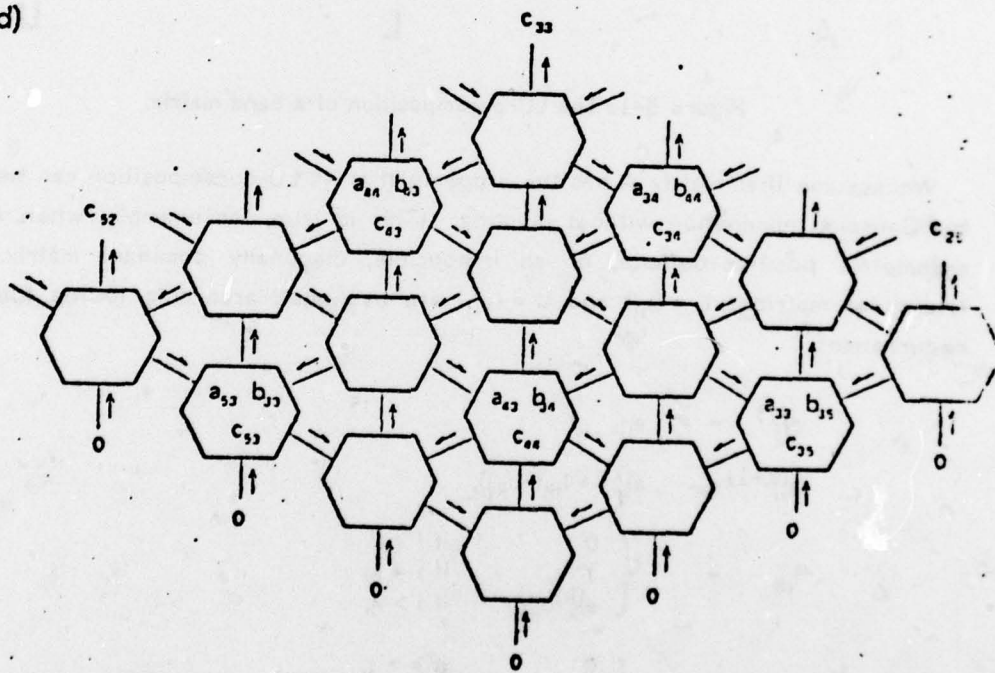


Figure 4-3: Four pulsations of the hexagonal systolic array in Figure 4-2.



## 5. The LU-Decomposition of a Matrix on a Hexagonal Systolic Array

The problem of factoring a matrix  $A$  into lower and upper triangular matrices  $L$  and  $U$  is called LU-decomposition. Figure 5-1 illustrates the LU-decomposition of a band matrix with  $p = 4$  and  $q = 4$ . Once the  $L$  and  $U$  factors are known, it is relatively easy to invert  $A$  or solve the linear system  $Ax = b$ . We deal with the latter problem in section 6. This section describes a hexagonal systolic array for computing LU-decompositions.

$$\begin{bmatrix}
 a_{11} & a_{12} & a_{13} & a_{14} & 0 \\
 a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\
 a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\
 a_{41} & a_{42} & a_{43} & & \\
 & a_{52} & a_{53} & & \\
 0 & & & & 
 \end{bmatrix}
 =
 \begin{bmatrix}
 1 & & & & \\
 l_{21} & 1 & & & \\
 l_{31} & l_{32} & 1 & & \\
 l_{41} & l_{42} & l_{43} & 1 & \\
 & l_{52} & l_{53} & & 1 \\
 0 & & & & 
 \end{bmatrix}
 \begin{bmatrix}
 u_{11} & u_{12} & u_{13} & u_{14} & 0 \\
 & u_{22} & u_{23} & u_{24} & u_{25} \\
 & & u_{33} & u_{34} & u_{35} \\
 & & & 0 & \\
 & & & & \\
 & & & & 
 \end{bmatrix}$$

$A$ 
 $L$ 
 $U$

Figure 5-1: The LU-decomposition of a band matrix.

We assume that matrix  $A$  has the property that its LU-decomposition can be done by Gaussian elimination without pivoting. (This is true, for example, when  $A$  is a symmetric positive-definite, or an irreducible, diagonally dominant matrix.) The triangular matrices  $L = (l_{ij})$  and  $U = (u_{ij})$  are evaluated according to the following recurrences.

$$\begin{aligned}
 a_{ij}^{(1)} &= a_{ij}, \\
 a_{ij}^{(k+1)} &= a_{ij}^{(k)} + l_{ik}(-u_{kj}), \\
 l_{ik} &= \begin{cases} 0 & \text{if } i < k, \\ 1 & \text{if } i = k, \\ a_{ik}^{(k)} u_{kk}^{-1} & \text{if } i > k, \end{cases} \\
 u_{kj} &= \begin{cases} 0 & \text{if } k > j, \\ a_{kj}^{(k)} & \text{if } k \leq j. \end{cases}
 \end{aligned}$$

We show that the evaluation of these recurrences can be pipelined on a



hex-connected systolic array of hex-connected processors. A global view of this pipelined computation is shown in Figure 5-2 for the LU-decomposition problem depicted in Figure 5-1. The systolic array in Figure 5-2 is constructed as follows. The processors below the upper boundaries are the standard type (b) inner product step processors and are hex-connected exactly same as the matrix multiplication network presented in Section 4. The processor at the top, denoted by a circle, is a special processor. It computes the reciprocal of its input and pumps the result southwest, and also pumps the same input northward unchanged. The other processors on the upper boundaries are again type (b) inner product step processors, but their orientation is changed: the ones on the upper left boundary are rotated 120 degrees clockwise; the ones on the upper right boundary are rotated 120 degrees counterclockwise.

The flow of data on the systolic array is indicated by arrows in the figure. As in the hexagonal systolic array for matrix multiplication, each processor only operates every third time pulse. Figure 5-3 illustrates four consecutive pulsations of the systolic array. Note that in the figure, because  $A$  is a band matrix with  $p = 4$  and  $q = 4$  we have that  $a_{i+3,i}^{(k)} = a_{i+3,i}$  and  $a_{i,i+3}^{(k)} = a_{i,i+3}$  for  $1 \leq k \leq i$  and  $i \geq 2$ . Thus  $a_{52}$ , for example, can be viewed as  $a_{52}^{(2)}$  when it enters the network.

There are several equivalent systolic arrays that reflect only minor changes to the network presented in this section. For example, the elements of  $L$  and  $U$  can be retrieved as output in a number of different ways. Also, the "-1" input to the network can be changed to a "+1" if the special processor at the top of the network computes minus the reciprocal of its input.

If  $A$  is an  $n \times n$  band matrix with band width  $w = p+q-1$ , a systolic array having no more than  $pq$  hex-connected processors can compute the LU-decomposition of  $A$  in  $3n+\min(p,q)$  units of time. If  $A$  is an  $n \times n$  dense matrix, this means that  $n^2$  hex-connected processors can compute the  $L$  and  $U$  matrices in  $4n$  units of time which includes I/O time.

The remarkable fact that the matrix multiplication network forms a major part of the LU-decomposition network is due to the similarity of their defining recurrences. In any row or column of the LU-decomposition systolic array, only one out of every three consecutive processors is active at a given time. As we observed for matrix multiplication, the number of processors can be reduced to about  $pq/3$ .

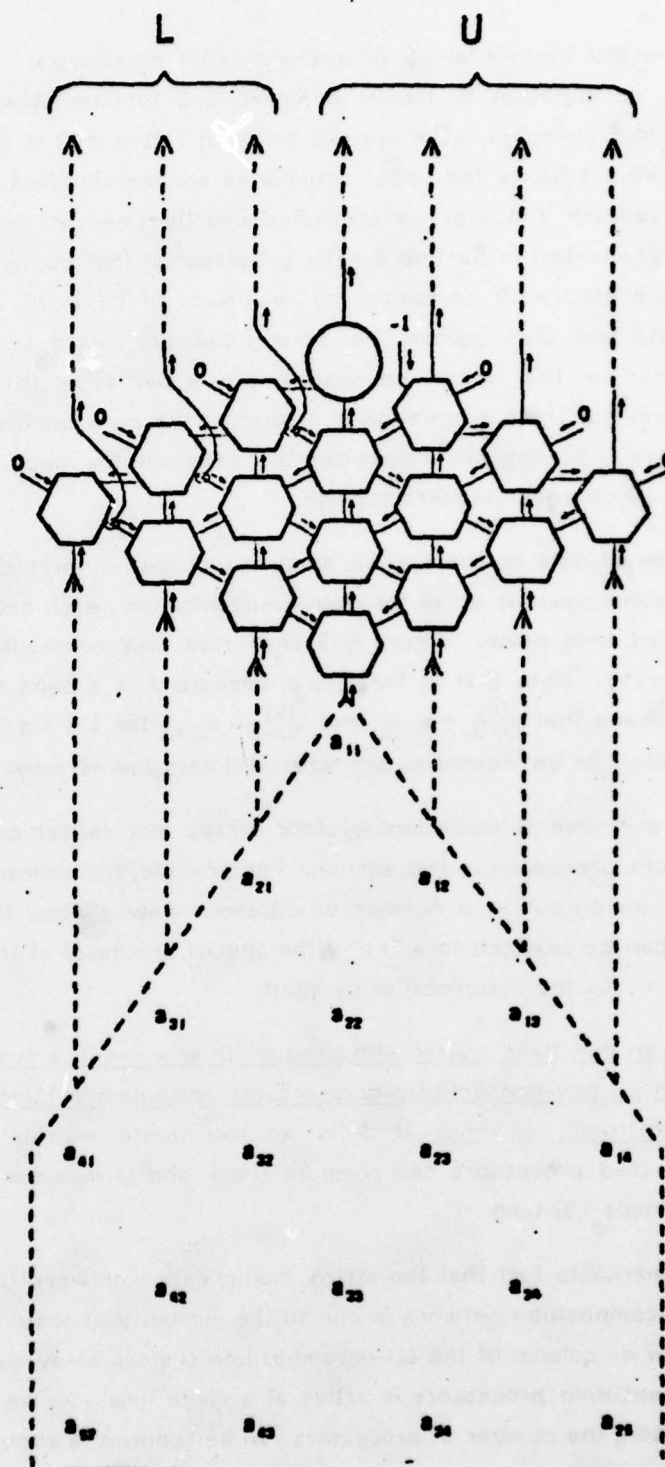
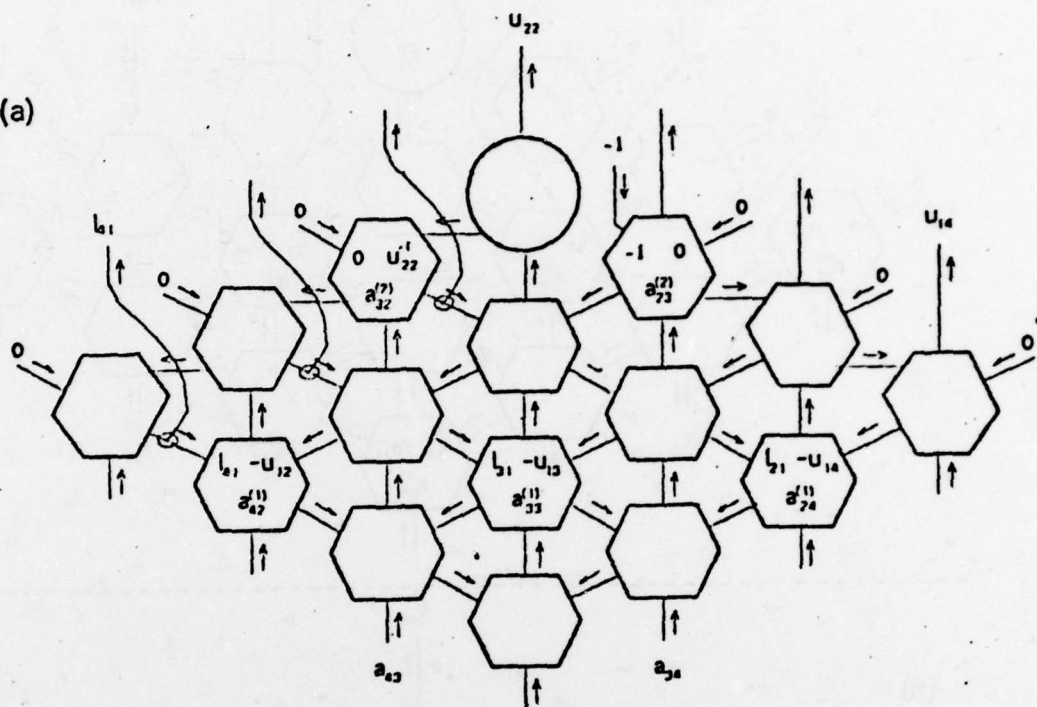
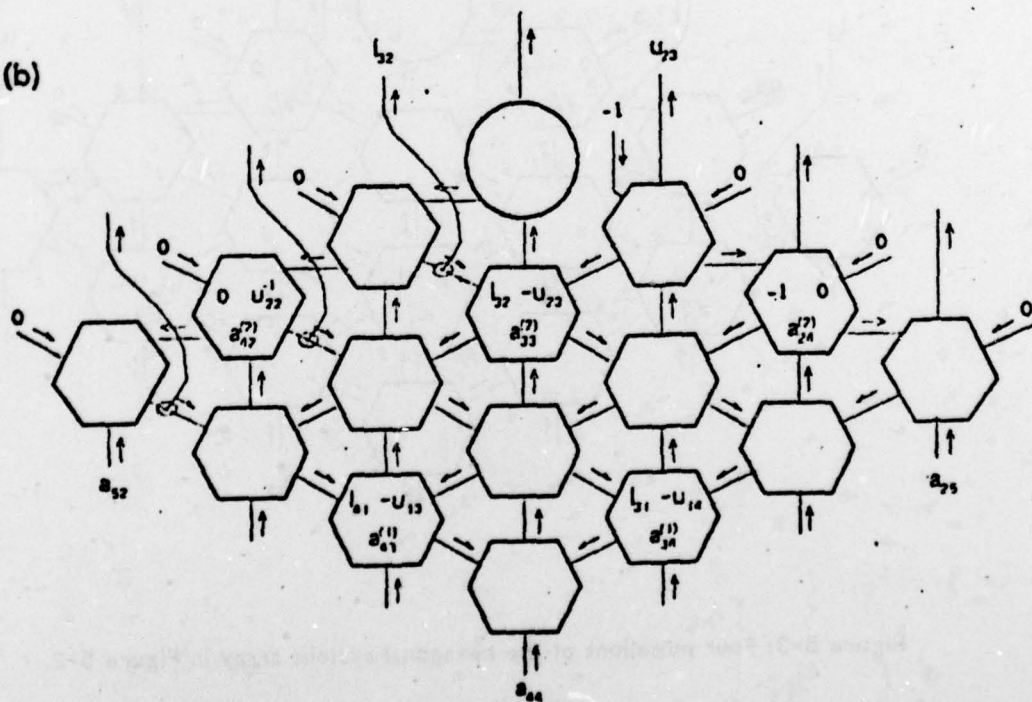


Figure 5-2: The hex-connected systolic array for pipelining the LU-decomposition of the band matrix in Figure 5-1.

(a)



(b)





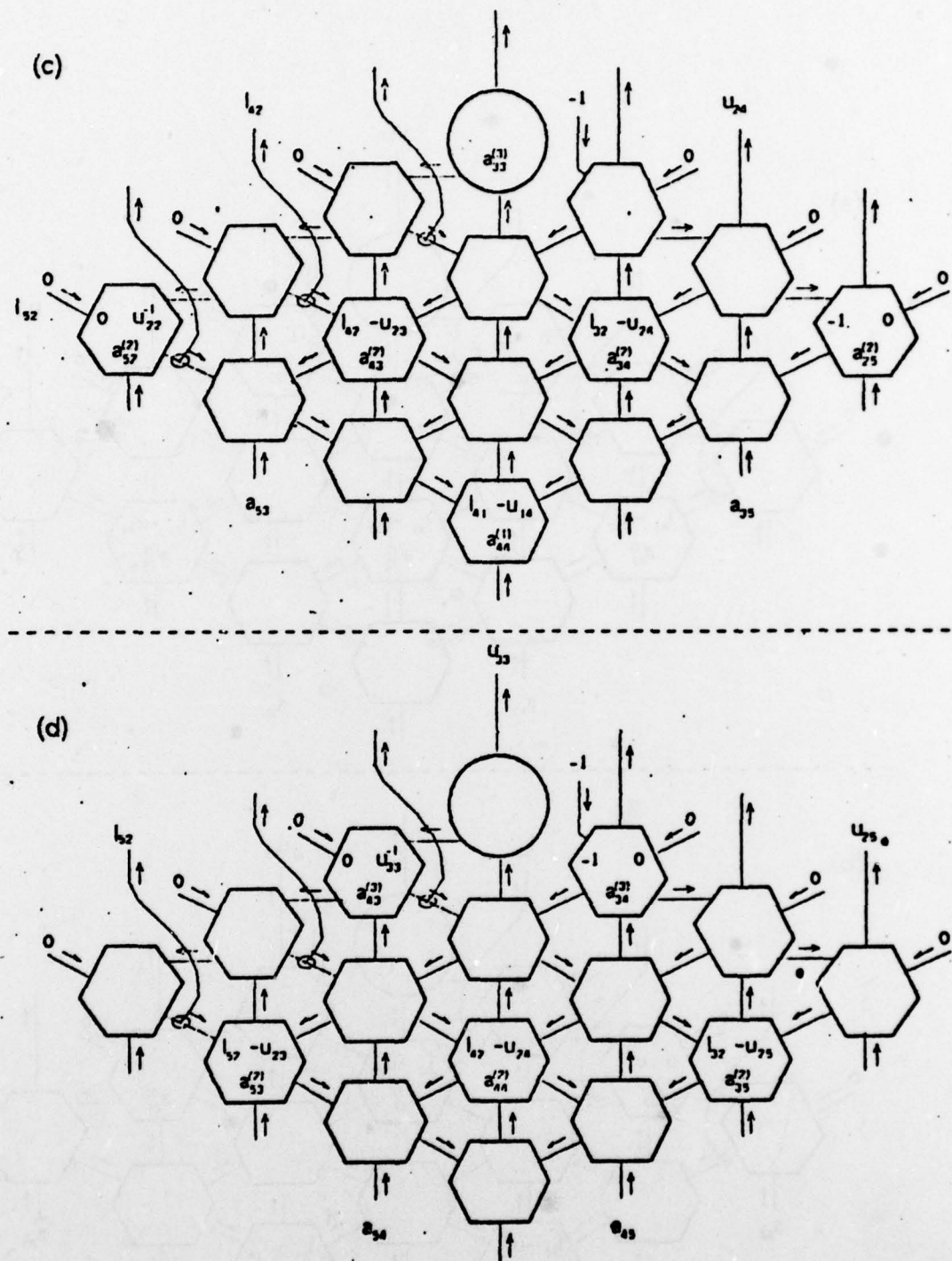


Figure 5-3: Four pulsations of the hexagonal systolic array in Figure 5-2.

## 6. Solving a Triangular Linear System on a Linear Systolic Array

Suppose that we want to solve a linear system  $Ax = b$ . Then after having done the LU-decomposition of  $A$  (e.g., by methods described in Section 5), we still have to solve two triangular linear systems  $Ly = b$  and  $Ux = y$ . This section concerns itself with the solution of triangular linear systems. An upper triangular linear system can always be rewritten as a lower triangular linear system. Without loss of generality, this section deals exclusively with lower triangular linear systems.

Let  $A = (a_{ij})$  be a nonsingular  $n \times n$  band lower triangular matrix. Suppose that  $A$  and an  $n$ -vector  $b = (b_1, \dots, b_n)^T$  are given. The problem is to compute  $x = (x_1, \dots, x_n)^T$  such that  $Ax = b$ . The vector  $x$  can be computed by forward substitution:

$$\begin{aligned} y_i^{(1)} &= 0, \\ y_i^{(k+1)} &= y_i^{(k)} + a_{ik}x_k, \\ x_i &= (b_i - y_i^{(i)})/a_{ii}. \end{aligned}$$

$$\begin{bmatrix} a_{11} & & & & & \\ a_{21} & a_{22} & & & & \\ a_{31} & a_{32} & a_{33} & & & \\ a_{41} & a_{42} & a_{43} & a_{44} & & \\ & a_{52} & a_{53} & a_{54} & a_{55} & \\ & & a_{63} & & & \\ & & & & & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ \vdots \\ \vdots \end{bmatrix}$$

$A \qquad x \qquad b$

Figure 6-1: The band (lower) triangular linear system where  $q = 4$ .

Suppose that  $A$  is a band matrix with band width  $w = q$ . (See Figure 6-1 for the case when  $q = 4$ .) Then the above recurrences can be evaluated by a systolic array similar to that used for band matrix-vector multiplication in Section 3. (Observe the similarity of the defining recurrences for these two problems.) We illustrate our

result by considering the linear system problem in Figure 6-1. For this case, the systolic array is described in Figure 6-2.

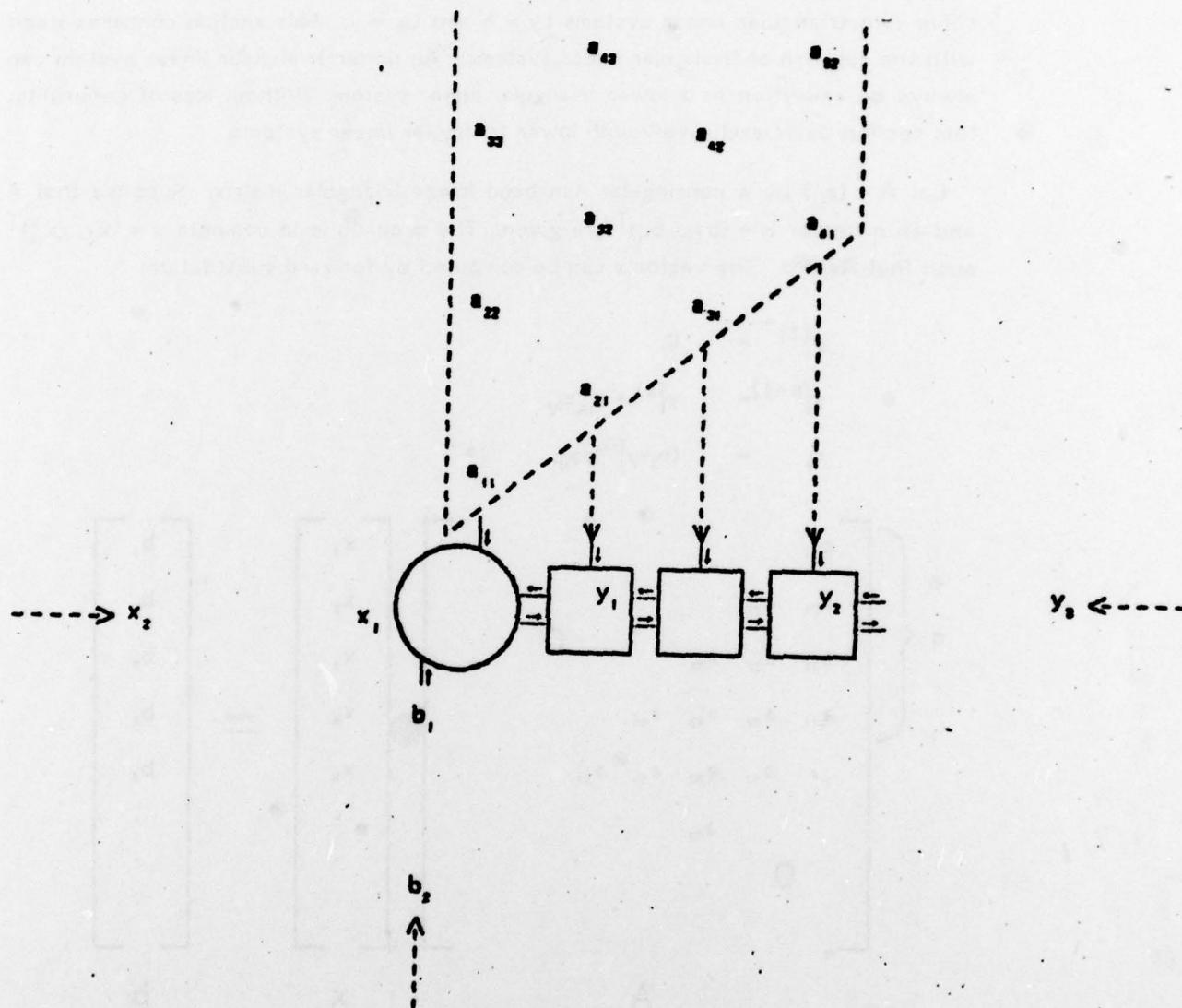


Figure 6-2: The linearly connected systolic array for solving the triangular linear system in Figure 6-1.

The  $y_i$ , which are initially zero, are forced leftward through the systolic array while the  $x_i$ ,  $a_{ij}$  and  $b_i$  are pumped as indicated in Figure 6-2. The left end processor is special in that it performs  $x_i \leftarrow (b_i - y_i)/a_{ii}$ . (In fact, the special processor introduced in section 5 to solve the LU-decomposition problem is a special case of



this more general processor.) Each  $y_i$  accumulates inner product terms in the rest of the processors as it moves to the left. At the time  $y_i$  reaches the left end processor it has the value  $a_{i1}x_1 + a_{i2}x_2 + \dots + a_{i,j-1}x_{j-1}$ , and, consequently, the  $x_j$  computed by  $x_j \leftarrow (b_j - y_i)/a_{ij}$  at the processor will have the correct value. Figure 6-3 demonstrates the first seven pulsations of the systolic array. From the figure one can check that the final values of  $x_1, x_2, x_3$  and  $x_4$  are all correct. With this systolic array we can solve an  $n \times n$  band triangular linear system with band width  $w = q$  in  $2n+q$  units of time. As we observed for the matrix-vector multiplication problem, the number of processors required by the array can be reduced to  $w/2$ .

## 7. Applications and Comments

### 7.1 Variants of the Systolic Array

If more information is available about the specific matrices involved, an optimized version of the systolic arrays presented above can be used. It is important that the reader understands the basic principles so that he can construct appropriate variants for his specific problems. No attempt is made here to list all the possible variants.

As pointed out in Section 1, although most of our illustrations are of band matrices, all the systolic arrays work for regular  $n \times n$  dense matrices. In this case the band width of the matrix is  $w = 2n-1$ . If the band width of a matrix is so large that it requires more processors than a given array provides, then one should decompose the matrix and solve each subproblem on the network. Thus, for example, the matrix multiplication of two  $n \times n$  matrices or the LU-decomposition of an  $n \times n$  matrix can be done in  $O(n^3/k^2)$  time on a  $k \times k$  systolic array.

One can often reduce the number of processors required by a systolic array if the matrix is known to be sparse or symmetric. For example, the matrices arising from a set of finite differences or finite elements approximations to differential equations are usually "sparse band matrices". These are band matrices whose nonzero entries appear only in a few of those lines in the band which are parallel to the diagonal. In this case by introducing proper delays to each processor for shifting its data to its neighbors, the number of processors required by the systolic array in Section 3 can be reduced to the number of those diagonals which contain nonzero entries. This variant is useful for performing iterative methods involving sparse band matrices. Another example concerns the LU-decomposition problem considered in Section 5. If matrix  $A$  is symmetric positive definite, then it is possible to use only the left portion of the hex-connected network, since in this case  $U$  is simply  $DL^T$  where  $D$  is the



Pulse Number	Configuration	Comments
0		$y_1$ enters processor 4.
1		$y_1$ moves left one position.
2		$y_2$ enters processor 4.
3		$x_1 = (b_1 - y_1)/a_{11}$ . ( $x_1 = b_1/a_{11}$ , since $y_1 = 0$ .)
4		$y_2 = a_{21} x_1$ .
5		$x_2 = (b_2 - y_2)/a_{22}$ . $y_3 = a_{31} x_1$ .
6		$y_3 = a_{31} x_1 + a_{32} x_2$ . $y_4 = a_{41} x_1$ .
7		$x_1$ is pumped out. $x_3 = (b_3 - y_3)/a_{33}$ . $y_4 = a_{41} x_1 + a_{42} x_2$ .
8		$y_4 = a_{41} x_1 + a_{42} x_2 + a_{43} x_3$ . $y_5 = a_{52} x_2$ .
9		$x_2$ is pumped out. $x_4 = (b_4 - y_4)/a_{44}$ . $y_5 = a_{52} x_2 + a_{53} x_3$ .

Figure 6-3: Solving a lower band triangular system.

diagonal matrix  $(a_{kk}^{(k)})$ .

The optimal choice of the size of the systolic network to solve a particular problem depends upon not only the problem but also the memory bandwidth to the host computer. For achieving high performance, it is desirable to have as many processors as possible in the network, provided they can all be kept busy doing useful computations.

It is possible to use our systolic arrays to solve some nonnumerical problems when appropriate interpretations are given to the addition (+) and multiplication (x) operations. For example, some pattern matching problems can be viewed as matrix problems with comparison and Boolean operations. It is possible to store a dynamically changing data structure in a systolic array so that an order statistic can always be determined in constant time. We shall report these results in a future paper. It can be instructive to view the + and x operations as operations in an abstract algebraic structure such as a semiring and then to examine how our results hold in such an abstract setting.

## 7.2 Convolution, Filter, and Discrete Fourier Transform

There are a number of important problems which can be formulated as matrix-vector multiplication problems and thus can be solved rapidly by the systolic array in Section 3. The problems of computing convolutions, finite impulse response (FIR) filters, and discrete Fourier transforms are such examples. If a matrix has the property that the entries on any line parallel to the diagonal are all the same, then the matrix is a Toeplitz matrix. The convolution problem is simply the matrix-vector multiplication where the matrix is a triangular Toeplitz matrix (see Figure 7-1).

A p-tap FIR filter can be viewed as a matrix-vector multiplication where the matrix is a band upper triangular Toeplitz matrix with band width  $w = p$ . Figure 7-2 represents the computation of a 4-tap filter.

On the other hand, an n-point discrete Fourier transform is the matrix-vector multiplication, where the  $(i,j)$  entry of the matrix is  $\omega^{(i-1)(j-1)}$  and  $\omega$  is a primitive  $n^{\text{th}}$  root of unity. (See Figure 7-3).

Therefore using a linearly connected systolic array of size  $n$  both the convolution of two  $n$ -vectors and the  $n$ -point discrete Fourier transform can be computed in  $O(n)$  units of time, rather than  $O(n \log n)$  as required by the sequential FFT algorithm. Moreover, note that for the convolution and filter problems each processor has to receive an entry of the matrix only once, and this entry can be shipped to the

$$\begin{bmatrix}
 a_1 & & & & & \\
 a_2 & a_1 & & & & \\
 a_3 & a_2 & a_1 & & & \\
 a_4 & a_3 & a_2 & a_1 & & \\
 a_5 & a_4 & a_3 & a_2 & a_1 & \\
 & \cdot & \cdot & \cdot & & \\
 & & & & & \cdot \\
 & & & & & \cdot \\
 & & & & & \cdot
 \end{bmatrix}
 \begin{bmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 \cdot \\
 \cdot \\
 \cdot
 \end{bmatrix}
 =
 \begin{bmatrix}
 b_1 \\
 b_2 \\
 b_3 \\
 b_4 \\
 b_5 \\
 \cdot \\
 \cdot \\
 \cdot
 \end{bmatrix}$$

Figure 7-1: The convolution of vectors  $a$  and  $x$ .

$$\begin{bmatrix}
 a_1 & a_2 & a_3 & a_4 & & & \\
 & a_1 & a_2 & a_3 & a_4 & & \\
 & & a_1 & a_2 & a_3 & a_4 & \\
 & & & a_1 & a_2 & a_3 & a_4 \\
 & & & & \cdot & & \\
 & & & & & \cdot & \\
 & & & & & & \cdot \\
 & & & & & & \cdot \\
 & & & & & & \cdot \\
 & & & & & & \cdot
 \end{bmatrix}
 \begin{bmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 \cdot \\
 \cdot \\
 \cdot \\
 \cdot
 \end{bmatrix}
 =
 \begin{bmatrix}
 y_1 \\
 y_2 \\
 y_3 \\
 y_4 \\
 \cdot \\
 \cdot \\
 \cdot \\
 \cdot
 \end{bmatrix}$$

$A$ 
 $x$ 
 $y$

Figure 7-2: A 4-tap FIR filter with coefficients  $a_1$ ,  $a_2$ ,  $a_3$ , and  $a_4$ .



$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 \\ 1 & \omega^2 & \omega^4 & \omega^8 & \omega^9 \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

Figure 7-3: The discrete Fourier transform of vector  $x$ .

processor through horizontal connections and stay in the processor during the rest of the computation. For the discrete Fourier transform problem each processor can in fact generate on-the-fly the powers of  $\omega$  it requires. As a result, for these three problems it is not necessary for each processor in the network to have the external input connection on the top of the processor, as depicted in Figure 3-2.

In the following we describe how the powers of  $\omega$  can be generated on-the-fly during the process of computing an  $n$ -point discrete Fourier transform. The requirement is that if a processor is  $i$  units apart from the middle processor then at time  $i + 2j$  the processor must have the value of  $\omega^{j^2 + ij}$  for all  $i, j$ . This requirement can be fulfilled by using the algorithm below. We assume that each processor has one additional register  $R_t$ . All processors except the middle one perform the following operations in each step, but for odd (respectively, even) numbered time steps only processors which are odd (even) units apart from the middle processor are activated. For all processors except the middle one the contents of both  $R_A$  and  $R_t$  are initially zero.

1. *Shift*. If the processor is in the left (respectively, right) hand side of the middle processor then
  - $R_A$  gets the contents of register  $R_A$  from the right (respectively, left) neighboring processor.
  - $R_t$  gets the contents of register  $R_t$  from the right (respectively, left) neighboring processor.

## 2. Multiply.

$$R_A \leftarrow R_A \times R_t.$$

The middle processor is special; it performs the following operations at every even numbered time step. For this processor the contents of both  $R_A$  and  $R_t$  are initially one.

$$1. R_A \leftarrow R_A \times R_t^2 \times \omega.$$

$$2. R_t \leftarrow R_t \times \omega.$$

## 7.3 The Common Memory Access Pattern

Note that all the systolic arrays given in this paper store and retrieve elements of the matrix in the same order.. (See Figures 3-2, 4-2, 5-2, and 6-2.) Therefore, we recommend that matrices be always arranged in memory according to this particular ordering so that they can be accessed efficiently by any of the systolic structures.

## 7.4 The Pivoting Problem, and Orthogonal Factorization

In section 5 we assume that the matrix  $A$  has the property that there is no need of using pivoting when Gaussian elimination is applied to  $A$ . What should one do if  $A$  does not have this nice property? (Note that Gaussian elimination becomes very inefficient on mesh-connect processors if pivoting is necessary.) This question motivated us to consider Givens' transformation (see, for example, Hammering [1974]) for triangularizing a matrix, which is known to be a numerically stable method. It turns out that, like Gaussian elimination without pivoting, the orthogonal factorization based on Givens' transformation can be implemented naturally on mesh-connected processors, although a pipelined systolic array implementation appears to be more complex. Our results on Givens' transformation will be reported in another paper. (Sameh and Kuck [1978] considers parallel linear system solvers based on Givens' transformation, but they do not give solutions to the processor communication problem considered in this paper.)

## 8. Concluding Remarks

Systolic structures provide a model of computation for studying parallel algorithms for VLSI. The model takes into account issues such as I/O, control, and interprocessor communication. In a systolic system pipelining can overlap I/O with

computation to ensure high throughput. Since loading of data into the network occurs naturally as computation proceeds, no extra control logic is required. Nor is initialization logic needed. Communication among processors is through fixed data paths. For a low cost and high performance implementation in VLSI (or even printed circuit technology), it is desirable that these paths have simple and regular geometries. These reasons make systolic arrays considered in this paper especially attractive. Indeed, interconnection structures other than arrays exist which satisfy these constraints. Future work will examine some of these connection schemes and demonstrate that systolic systems generalize beyond simple cellular structures.

We have discovered that some data flow patterns are fundamental in matrix computations. For example, the two-way flow on the linearly connected network is common to both matrix-vector multiplication and solution of triangular linear systems (Sections 3 and 6), and the three-way flow on the hexagonally mesh-connected network is common to both matrix multiplication and LU-decomposition (Sections 4 and 5). A practical implication of this fact is that one systolic device may be used for solving many different problems. Moreover, we note that almost all the processors needed in any of these devices are the inner product step processor postulated in Section 2. A careful design of this processor is desirable since it is the work horse for all the devices presented.

Research in interconnection networks and algorithms has been frequently motivated by parallel array computers such as ILLIAC IV. (See, for example, Kuck [1968, 1977] and Stone [1975].) Although the results presented in this paper were motivated by the advance in VLSI, they reach beyond. The systolic arrays in this paper can be implemented as efficient algorithms on traditional parallel array machines.

For the important problem of solving a dense system of  $n$  linear equations in  $O(n)$  time on  $n^2$  mesh-connected processors, we have improved upon the recent results of Kant and Kimura [1978]. The basis of their results is an theorem on determinants which was known to J. Sylvester in 1851. Their algorithm requires that the matrix be "strongly nonsingular" in the sense that every square submatrix is nonsingular. It is sufficient for our algorithms that the matrix be symmetric positive-definite or irreducible diagonally dominant.

Hoare [1977], and Thurber and Wald [1975] describe some matrix multiplication algorithms on an orthogonally connected processor array. Unlike our results, their algorithms require that one or more of the three matrices involved in matrix multiplication stay in the array statically during the computation. This introduces



overheads in I/O time and control logic for loading the array with the static matrix. Our systolic array makes use of the hexagonal connections to pipeline all three matrices.

Processor communication will likely dominate the cost of parallel algorithms and systems. Communication paths inherently require more space and energy than processing elements do. We regard the problem of minimizing communication costs as fundamental, and we believe systolic structures provide models that can bridge the gap between theory and practice. Systolic arrays can be built in VLSI. Connected to a standard Von Neumann computer, a systolic device provides inexpensive but massive computation power.

#### REFERENCES

- Barnes et al. [1968] Barnes, G. H., Brown, R. M., Maso, K., Kuck, D. J., Slotnick, D. L., and Stokes, R. A., "The ILLIAC IV Computer," IEEE Transactions on Computers C-17 (1968), pp. 746-757.
- Blakeslee [1975] Blakeslee, Thomas R., Digital Design with Standard MSI and LSI, John Wiley & Sons, New York, 1975.
- Hammering [1974] Hammering, S., "A Note on Modifications to the Givens' Plane Rotation," J. Inst. Math. Appl. 13 (1974), pp. 215-218.
- Hoare [1977] Hoare, C. A. R., "Communicating Sequential Processes," Communications of the ACM 21 (1978), pp. 666-677.
- Kant and Kimura [1978] Kant, Rajani M. and Kimura, Takayuki, "Decentralized Parallel Algorithms for Matrix Computation," Proceedings of the Fifth Annual Symposium on Computer Architecture, Palo Alto, California, April 1978, pp. 96-100.
- Kuck [1968] Kuck, D. J., "ILLIAC IV Software and Application Programming," IEEE Transactions on Computers C-17 (1968), pp. 758-770.
- Kuck [1977] Kuck, D. J., "A Survey of Parallel Machine Organization and Programming," Computing Surveys 9 (1977), pp. 29-59.
- Kung and Leiserson [1978] Kung, H. T. and Leiserson, Charles E., "Systolic Array Apparatuses for Matrix Computations," U. S. Patent Application, Filed December 11, 1978.
- Mead and Conway [1978] Mead, C. A. and Conway, L. A., Introduction to VLSI Systems, 1978.
- Samah and Kuck [1978] Samah, A. H. and Kuck, D. J., "On Stable Parallel Linear System Solvers," Journal of the Association for Computing Machinery 25 (1978), pp. 81-91.



Stone [1975] Stone, Harold S., "Parallel Computations", Introduction to Computer Architecture, edited by H. S. Stone, Science Research Associates, Chicago, 1975, pp. 318-374.

Sutherland and Mead [1977] Sutherland, Ivan E. and Mead, Carver A., "Microelectronics and Computer Science," Scientific American 237 (1977), pp. 210-228.

Thurber and Wald [1975] Thurber, Kenneth J., and Wald, Leon D., "Associative and Parallel Processors," Computing Surveys 7 (1975), pp. 215-255.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) CMU-CS-79-103	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) SYSTOLIC ARRAYS FOR (VLSI)	5. TYPE OF REPORT & PERIOD COVERED (9) Interim rept's	
7. AUTHOR(s) (10) H.T. Kung and Charles E. Leiserson	6. PERFORMING ORG. REPORT NUMBER	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Department Pittsburgh, PA 15213	8. CONTRACT OR GRANT NUMBER(s) (15) N00014-76-C-0370 NSF-MCS 75-22255	
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same as above (12) 32 p	12. REPORT DATE (11) December 1978	
	13. NUMBER OF PAGES 32	
	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		

DD FORM 1473  
1 JAN 73EDITION OF 1 NOV 65 IS OBSOLETE  
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403 081

set